05/11/2023

Object-Oriented Programming – Second Midterm

(Open textbook, notes, compiler, 180 minutes)

Total Score **100%**

Name: _____________

ID: ________________

Grading rule

I. Your program must be **compiled without errors to earn credits for that problem.**
II. The corresponding outputs should be produced, even in terms of format.
III. If you cannot achieve the example output, you can write comments in your code to explain reasons why, and only then partial credits could be given.

1. **(25%)** Write a `Rectangle` class, which has private data members: `string color`, `double length` and `double width`. Please add required member functions.

You CANNOT change the following function.

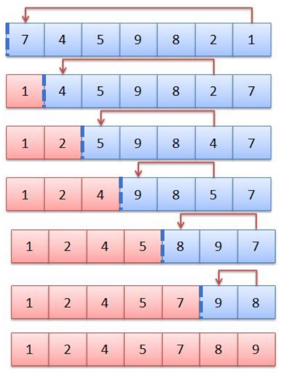
```
int main() {
  cout << "Start of main() "<< endl;
  cout << "1. ";
  Rectangle r1("Yellow",30, 40);
  r1.printInfo();

  cout << "2. ";
  Rectangle r2(r1);
  r2.printInfo();
  cout << "3. is r2 square? " << r2.isSquare() << endl;

  cout << "4. ";
  Rectangle r3;
  r3.printInfo();

  cout << "5. ";
  r2.changeColor("Green");
  r2.changeWidth(100);
  r2.changeLength(100);
  r3 = r2;
  r3.printInfo();
  cout << "6. is r3 square? " << r3.isSquare() << endl;
  cout << "End of main() "<< endl;
}
```

Output:

```
Start of main()
1. Color: Yellow, width: 30, length: 40
2. Color: Yellow, width: 30, length: 40
3. is r2 square? 0
4. Color: White, width: 1, length: 1
5. Color: Green, width: 100, length: 100
6. is r3 square? 1
End of main()
```

2. **(25%)** Please complete the `Vec` class. Two of its member functions are part of the selection sort algorithm, which is to sort a list of numbers into increasing order.

We separate the array into a sorted portion (left) and an unsorted portion (right). The selection sort algorithm repeatedly finds a minimum value in the unsorted portion. Once the minimum value in the unsorted portion is found, we switch (swap) positions of the min value and the left-most-value in the unsorted portion, and the unsorted portion shrinks its size by 1.



- Data members are: `v` the pointer to the integer array, and `n` the size of the array.
- The constructor reads in the input from the user.

```
Vec();
```

- The first member function is to find, from the `start` position, the min value from the integer array. The return value is the position of the min value in the array.

```
int findMin(int start);
```

- The second function is to switch two integers based on their positions in the array.

```
void swapPos(int pos1, int pos2);
```

- The third function is the overall logic of the selection sort. You will use the first two functions in this function to complete the selection sort algorithm.

```
void selectionSort();
```

- The fourth function is to display the vector.

```
void printOut();
```

- The `Vec` class has **NO** friend functions.
- You can assume input values will not exceed 100.0.

The following is the main function you **CANNOT change**.

```
int main(){
    Vec v;
    cout << "Original input: " << v;
    v.selectionSort();
    cout << "Final result: " << v;
}
```

Example:

```
Please input how many numbers you wish to sort: 7
Please input the numbers: 7 4 5 9 8 2 1
Original input: 7 4 5 9 8 2 1
Swaping values: (7, 1): 1 4 5 9 8 2 7
Swaping values: (4, 2): 1 2 5 9 8 4 7
Swaping values: (5, 4): 1 2 4 9 8 5 7
Swaping values: (9, 5): 1 2 4 5 8 9 7
Swaping values: (8, 7): 1 2 4 5 7 9 8
Swaping values: (9, 8): 1 2 4 5 7 8 9
Final result: 1 2 4 5 7 8 9
```

3. **(25%)** In this problem, you will implement a class called `SharedArray`. It has private members: `int size` and `int* data`. For any object created from this class, they all share the same copy (memory) of `data`. This class has a friend function for the "<<" operator. You should add required members such as data, constructors, operators and functions.

The following are functions you cannot change:

```
SharedArray create(){
     return SharedArray(5);
}

int main(){
    SharedArray m;
    cout << "before call to create()" << endl;
    m = create();
    m[0] = 5;
    cout << "m: " << m;
    const SharedArray n(m);
    m[0] = 1;
    m[2] = n[0];
    cout << "m: " << m;
    cout << "n: " << n;
    SharedArray o;
    o = m;
    cout << "o: " << o;
    SharedArray p = move(create());
    cout << "before returning from main" << endl;
    return 0;
}
```

Output:

```
Default Constructor is called
before call to create()
Constructor is called
Move Assignment
Destructor is called but data still in use by other object!
instances left: 1
m: 5 0 0 0 0
Copy Constructor is called -Shallow copy
m: 1 0 1 0 0
n: 1 0 1 0 0
Default Constructor is called
Copy Assignment
o: 1 0 1 0 0
Constructor is called
Move Constructor
Destructor is called but data still in use by other object!
instances left: 4
before returning from main
Destructor is called but data still in use by other object!
instances left: 3
Destructor is called but data still in use by other object!
instances left: 2
Destructor is called but data still in use by other object!
instances left: 1
Destructor is called and clean up is done!
instances left: 0
```

4. **(25%)** In this problem, you are implementing the `OOPStack` class. It is a template for the `T` class, and it has 3 private data members:
   1. `int size`: number of T objects stored in stack. The initial value is 0.
   2. `int capacity`: how many T objects can be stored (internal memory allocated for the container). The default value is 2.
   3. `T* vecPtr`: the pointer to the memory.

   You should add required members such as constructors, operators and functions.

   The following is the main you **cannot** change:

```
int main(){
   cout << "int case:" << endl;
   OOPStack<int> a;
   for(int i = 0; i < 10; i++){
      a.place(i*i);
   }
   cout << a.pop() << endl;
   cout << a.getSize() << endl;

   cout << "bool case:" << endl;
   OOPStack<bool> b(10); // constructor for capacity of 10
   for(int i = 0; i < 30; i++){
      b.place(i%3==0);
   }
   cout << b.pop() << endl;
   cout << b.pop() << endl;
   cout << b.pop() << endl;
}
```

   Output:

```
int case:
OOPStack capacity doubled to 4
OOPStack capacity doubled to 8
OOPStack capacity doubled to 16
81
9
bool case:
OOPStack capacity doubled to 20
OOPStack capacity doubled to 40
0
0
1
```